M&G No. 60001.0296US01/305244.01

STORING DOCUMENT HEADER AND FOOTER INFORMATION IN A MARKUP LANGUAGE DOCUMENT

Related Applications

This patent application is a continuation-in-part application under 35 United States Code § 120 of United States Patent Application No. 10/187,060 filed on June 28, 2002, which is incorporated herein by reference. An exemplary schema in accordance with the present invention is disclosed beginning on page 11 in an application entitled "Mixed Content Flexibility," Serial No. _________, Docket No. 60001.0275US01, filed December 2, 2003, which is hereby incorporated by reference in its entirety.

Background of the Invention

Markup Languages have attained wide popularity in recent years. One type of markup language, Extensible Markup Language (XML), is a universal language that provides a way to identify, exchange, and process various kinds of data. For example, XML is used to create documents that can be utilized by a variety of application programs. Elements of an XML file have an associated namespace and schema.

In XML, a namespace is a unique identifier for a collection of names that are used in XML documents as element types and attribute names. The name of a namespace is commonly used to uniquely identify each class of XML document. The unique namespaces differentiate markup elements that come from different sources and happen to have the same name.

XML Schemata provide a way to describe and validate data in an XML environment. A schema states what elements and attributes are used to describe content in an XML document, where each element is allowed, what types of text contents are allowed within it and which elements can appear within which other elements. The use of schemata ensures that the document is structured in a consistent manner. Schemata may be created by a user and generally supported by an associated markup language, such as XML. By using an XML editor, the user can manipulate the XML file and generate XML documents that adhere to the schema the user has created. XML documents may be created to adhere to one or more schemata.

The XML standard is considered by many as the ASCII format of the future, due to its expected pervasiveness throughout the hi-tech industry in the coming years. Recently, some word-processors have begun producing documents that are somewhat XML compatible. For example, some documents may be parsed using an application that understands XML. However, much of the functionality available in word processor documents is not currently available for XML documents.

Summary of the Invention

The present invention is generally directed towards a method for storing header and footer information in a markup language (ML) document such as an XML document. Headers and footers may be generally understood as simple "mini-documents" that show up at the top and/or bottom of each page in a particular section of a document. Headers and footers simplify functions such as page numbering and template functions for providing a logo on each page of a document.

More particularly, the present invention relates to representing header and footer information in ML so that applications capable of reading a given ML file format, but running in environments where the header and footer generation information has not been installed, are able to still render the header and footer structures. The ML document may be manipulated on a server or anywhere even when the application creating the ML document is not present. Header and footer information (i.e., properties) are saved in a markup language (ML) document without data loss, while allowing the header and footer structures to be parsed by ML-aware applications and to be read by ML programmers.

Brief Description of the Drawings

FIGURE 1 illustrates an exemplary computing device that may be used in one exemplary embodiment of the present invention;

FIGURE 2 is a block diagram illustrating an exemplary environment for practicing the present invention;

FIGURE 3 illustrates an exemplary portion of an ML file that provides representation of an exemplary header within the ML file;

FIGURE 4 illustrates an exemplary portion of an ML file that provides representation of an exemplary header within the ML file; and

FIGURE 5 shows an exemplary flow diagram for representing header and footer structures in a ML document, in accordance with aspects of the invention.

Detailed Description of the Preferred Embodiment

Throughout the specification and claims, the following terms take the meanings explicitly associated herein, unless the context clearly dictates otherwise.

The terms "markup language" or "ML" refer to a language for special codes within a document that specify how parts of the document are to be interpreted by an application. In a word-processor file, the markup language specifies how the text is to be formatted or laid out, whereas in a particular customer schema, the ML tends to specify the text's meaning according to that customer's wishes (e.g., customerName, address, etc). The ML is typically supported by a word-processor and may adhere to the rules of other markup languages, such as XML, while creating further rules of its own.

The term "element" refers to the basic unit of an ML document. The element may contain attributes, other elements, text, and other building blocks for an ML document.

The term "tag" refers to a command inserted in a document that delineates elements within an ML document. Each element can have no more than two tags: the start tag and the end tag. It is possible to have an empty element (with no content) in which case one tag is allowed.

The content between the tags is considered the element's "children" (or descendants). Hence, other elements embedded in the element's content are called "child elements" or "child nodes" or the element. Text embedded directly in the content of the element is considered the element's "child text nodes". Together, the child elements and the text within an element constitute that element's "content".

The term "attribute" refers to an additional property set to a particular value and associated with the element. Elements may have an arbitrary number of attribute settings associated with them, including none. Attributes are used to associate additional information with an element that will not contain additional elements, or be treated as a text node.

Illustrative Operating Environment

With reference to FIGURE 1, one exemplary system for implementing the invention includes a computing device, such as computing device **100**. In a very basic configuration, computing device **100** typically includes at least one processing unit **102** and system memory **104**. Depending on the exact configuration and type of computing device, system memory **104** may be volatile (such as RAM), non-volatile (such as ROM, flash memory, etc.) or some combination of the two. System memory **104** typically includes an operating system **105**, one or more applications **106**, and may include program data **107**. In one embodiment, application **106** may include a word-processor application **120** that further includes header and footer structures **122**. This basic configuration is illustrated in FIGURE 1 by those components within dashed line **108**.

Computing device **100** may have additional features or functionality. For example, computing device **100** may also include additional data storage devices (removable and/or non-removable) such as, for example, magnetic disks, optical disks, or tape. Such additional storage is illustrated in FIGURE 1 by removable storage **109** and non-removable storage **110**. Computer storage media may include volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information, such as computer readable instructions, data structures, program modules, or other data. System memory **104**, removable storage **109** and non-removable storage **110** are all examples of computer storage media. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computing device **100**. Any such computer storage media may be part of device **100**. Computing device **100** may also have input device(s) **112** such as keyboard, mouse, pen, voice input device, touch input device, etc. Output device(s) **114** such as a display, speakers, printer, etc. may also be included. These devices are well know in the art and need not be discussed at length here.

Computing device **100** may also contain communication connections **116** that allow the device to communicate with other computing devices **118**, such as over a network. Communication connection **116** is one example of communication media. Communication media may typically be embodied by computer readable instructions, data structures, program modules, or other data in a modulated data signal, such as a carrier wave or other transport mechanism, and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. The term computer readable media as used herein includes both storage media and communication media.

Generally, the present invention is directed at representing header and footer structures in an ML document. The ML document may be read by applications that do not share the same schema that created the document.

FIGURE 2 is a block diagram illustrating an exemplary environment for practicing the present invention. The exemplary environment shown in FIGURE 2 is a word-processor environment **200** that includes word-processor **120**, ML file **210**, ML Schema **215**, and ML validation engine **225**.

In one embodiment, word-processor **120** has its own namespace or namespaces and a schema, or a set of schemas, that is defined for use with documents associated with word-processor **120**. The set of tags and attributes defined by the schema for word-processor **120** define the format of a document to such an extent that it is referred to as its own native ML. Word-processor **120** internally validates ML file **210**. When validated, the ML elements are examined as to whether they conform to the ML schema **215**. A schema states what tags and attributes are used to describe content in an ML document, where each tag is allowed, and which tags can appear within other tags, ensuring that the documentation is structured the same way. Accordingly, ML **210** is valid when structured as set forth in arbitrary ML schema **215**.

ML validation engine **225** operates similarly to other available validation engines for ML documents. ML validation engine **225** evaluates ML that is in the format of the ML validation engine **225**. For example, XML elements are forwarded to an XML validation engine. In one embodiment, a greater number of validation engines may be associated with word-processor **120** for validating a greater number of ML formats.

Storing Header and Footer Information in a Markup Language Document

The present invention generally provides a method to represent an application's header and footer information in markup language (ML) such as XML. The header and footer structures may be parsed by applications that understand the markup other than the application that generated the ML file.

Headers and footers are used for a number of different applications. In one example, a footer is used to show the page number at the bottom of each page (like in this actual document), the author of the document may choose to create a footer. The footer is a "mini-document" that appears at the bottom of each page. A field is placed in the generated mini-document that instructs the footer to display the current page. With the use of the footer, a single mini-document is generated, but each time the mini-document appears, it shows the page number of that page. In another example, an author chooses to include the language "confidential" at the top of each page. To generate the "confidential" language on each page, a header is generated. The "mini-document" that is the header, therefore includes the text "confidential". As a result, each page includes a view of this mini-document such that the top of each page reads "confidential".

FIGURE 3 illustrates an exemplary portion of an ML file that provides representation of an exemplary header within the ML file, in accordance with aspects of the present invention. The header example shown displays the text "Header Info" at the top of the odd pages in the document produced from the ML file.

Analyzing the example shown in FIGURE 3, the header type is designated as "odd". Accordingly, the text "Header Info" is displayed on the odd pages of the document produced from the ML file. The portion of the ML file shown also includes a designation of the size of the pages, the margins associated with each page, and the positioning of the header within each page. In further embodiments, a variety of fields other than the text shown may be associated with the header. Accordingly, the properties and functionality of the header stored in the ML file is not limited to the example shown, and a number of variations for a header are available.

Headers and footers are specific to a particular section. Many documents only consist of one section, but that is not always the case. Because documents may include more than one section, the header and footer information is stored with the section properties. There are two elements in the section properties tag: <hdr> and <ftr>. These are option elements, and they are each of type hdrElt and ftrElt respectively. In one embodiment, the types of option elements are substantially the same, but the schema defines them separately.

The hdrElt shown in the example ML file includes the following:

ref = "aml:annotation"	**xmlns = "http://schemas.microsoft.com/aml/2001/core"**	
cfChunk	**[cfChunkElt]**	*'Context-Free' Chunk -- allows inline definition of stylesheet, font, and list items*

p	**[pElt]**	*Paragraph element; analogous to HTML <p> tag*
tbl	**[tblElt]**	*Table element; analogous to HTML <table>*

Table 1

In one embodiment, the definitions for the header and footer elements are substantially the same as those of the body element of the document. Accordingly, headers and footers may be described as “mini-documents” due to their similarity to the body element.

In one embodiment, a type attribute is associated with the header or footer. The type attribute provides functionality for multiple headers & footers to be associated with any one section. In one example, the type attribute may be set up such that a header only appears on even pages; only appears on odd pages; or only appears on the first page.

FIGURE 4 illustrates an exemplary portion of an ML file that provides representation of an exemplary header within the ML file, in accordance with aspects of the present invention. The footer example shown displays the text "Footer Info" at the bottom of the odd pages within the document produced from the ML file.

Analyzing the example shown in FIGURE 4, the footer type is designated as "odd". Accordingly, the text "Footer Info" is displayed on the odd pages of the document produced from the ML file. The portion of the ML file shown also includes a designation of the size of the pages, the margins associated with each page, and the positioning of the footer within each page. The example of the footer shown in FIGURE 4, is similar to the example for the header shown in FIGURE 3. In further embodiments, a variety of fields other than the text shown may be associated with the footer. Accordingly, the properties and functionality of the footer stored in the ML file is not limited to the example shown, and a number of variations for a footer are available.

The following is an exemplary portion of schema that includes the section properties element as well as the schema for generating the header and footer mini-documents along with the type attribute, in accordance with aspects of the present invention:

Schema definition of Section Properties element:

```
<xsd:complexType name="sectPrElt">
        <xsd:sequence>
                <xsd:element name="hdr" type="hdrElt" minOccurs="0"
maxOccurs="3">
                        <xsd:annotation>
                                <xsd:documentation>Headers that appear at the
top page in this section.</xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="ftr" type="ftrElt" minOccurs="0"
maxOccurs="3">
                        <xsd:annotation>
                                <xsd:documentation>Footers that appear at the top
of the page in this section.</xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
                <xsd:element name="footnotePr" type="ftnEdnPropsElt"
minOccurs="0">
                        <xsd:annotation>
                                <xsd:documentation>Footnote properties for this
section.</xsd:documentation>
                        </xsd:annotation>
                </xsd:element>
```

```
                    <xsd:element name="endnotePr" type="ftnEdnPropsElt"
minOccurs="0">
                        <xsd:annotation>
                            <xsd:documentation>Endnote properties for this
section.</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="type" type="sectTypeElt" minOccurs="0">
                        <xsd:annotation>
                            <xsd:documentation>Section
type.</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="pgSz" type="pageSzType"
minOccurs="0">
                        <xsd:annotation>
                            <xsd:documentation>Specifies the size and
orientation of this page.</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="pgMar" type="pageMarType"
minOccurs="0">
                        <xsd:annotation>
                            <xsd:documentation>Specifies the page
margins.</xsd:documentation>
                        </xsd:annotation>
                    </xsd:element>
                    <xsd:element name="paperSrc" type="paperSourceType"
minOccurs="0">
                        <xsd:annotation>
```

```
                                        <xsd:documentation>Specifies where the paper is
located in the printer.</xsd:documentation>
                                </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="pgBorders" type="pageBordersType"
minOccurs="0">
                                <xsd:annotation>
                                        <xsd:documentation>Specifies the page
borders.</xsd:documentation>
                                </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="lnNumType" type="lineNumberType"
minOccurs="0">
                                <xsd:annotation>
                                        <xsd:documentation>Specifies the line
numbering.</xsd:documentation>
                                </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="pgNumType" type="pageNumberType"
minOccurs="0">
                                <xsd:annotation>
                                        <xsd:documentation>Specifies the page
numbering options.</xsd:documentation>
                                </xsd:annotation>
                        </xsd:element>
                        <xsd:element name="cols" type="columnsType"
minOccurs="0">
                                <xsd:annotation>
                                        <xsd:documentation>Specifies the column
properties for this section.</xsd:documentation>
```

```
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="formProt" type="onOffProperty"
minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>Turns protection on for this
section alone.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="vAlign" type="verticalJustificationType"
minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>Sets alignment for text
vertically between the top and bottom margins.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="noEndnote" type="onOffProperty"
minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>Suppresses endnotes that
would ordinarily appear at the end of this section.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="titlePg" type="onOffProperty"
minOccurs="0">
                    <xsd:annotation>
                        <xsd:documentation>The first page of this section
is different and will have different headers/footers.</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
```

```
				<xsd:element name="textFlow" type="textDirectionProperty"
minOccurs="0">
					<xsd:annotation>
						<xsd:documentation>Specifies text
flow.</xsd:documentation>
					</xsd:annotation>
				</xsd:element>
				<xsd:element name="bidi" type="onOffProperty"
minOccurs="0">
					<xsd:annotation>
						<xsd:documentation>This section contains bi-
directional (Complex Scripts) text.</xsd:documentation>
					</xsd:annotation>
				</xsd:element>
				<xsd:element name="rtlGutter" type="onOffProperty"
minOccurs="0">
					<xsd:annotation>
						<xsd:documentation>Positions the gutter at the
right of the document.</xsd:documentation>
					</xsd:annotation>
				</xsd:element>
				<xsd:element name="docGrid" type="docGridType"
minOccurs="0">
					<xsd:annotation>
						<xsd:documentation>Specifies the document
grid.</xsd:documentation>
					</xsd:annotation>
				</xsd:element>
				<xsd:element ref="aml:annotation" minOccurs="0"
maxOccurs="1">
```

```
                                        <xsd:annotation>
                                                <xsd:documentation>Revision marking for the
section properties.</xsd:documentation>
                                        </xsd:annotation>
                                </xsd:element>
                        </xsd:sequence>
</xsd:complexType>
```

Schema definition of Header element:

```
<xsd:complexType name="hdrElt">
                <xsd:annotation>
                        <xsd:documentation>Headers are areas at in the top margin of
each page in the current section.</xsd:documentation>
                </xsd:annotation>
                <xsd:sequence>
                        <xsd:choice maxOccurs="unbounded">
                                <xsd:element ref="aml:annotation" minOccurs="0"
maxOccurs="unbounded"></xsd:element>
                                <xsd:element name="cfChunk" type="cfChunkElt"
minOccurs="0" maxOccurs="unbounded">
                                        <xsd:annotation>
                                                <xsd:documentation>'Context-Free' Chunk
-- allows inline definition of stylesheet, font, and list items</xsd:documentation>
                                        </xsd:annotation>
                                </xsd:element>
                                <xsd:element name="p" type="pElt" minOccurs="0"
maxOccurs="unbounded">
                                        <xsd:annotation>
                                        <xsd:documentation>Paragraph element;
analogous to HTML &lt;p&gt; tag</xsd:documentation>
                                    </xsd:annotation>
                                </xsd:element>
                                <xsd:element name="tbl" type="tblElt" minOccurs="0"
maxOccurs="unbounded">
                                    <xsd:annotation>
                                        <xsd:documentation>Table element;
analogous to HTML &lt;table&gt;</xsd:documentation>
                                    </xsd:annotation>
                                </xsd:element>
                            </xsd:choice>
                        </xsd:sequence>
                        <xsd:attribute name="type" type="hdrValue" use="required">
                            <xsd:annotation>
                                <xsd:documentation>Specifies the header
type.</xsd:documentation>
                            </xsd:annotation>
                        </xsd:attribute>
</xsd:complexType>
```

Schema definition of type attribute:

```
<xsd:simpleType name="hdrValue">
                        <xsd:annotation>
                            <xsd:documentation>Specifies the header
type.</xsd:documentation>
                        </xsd:annotation>
                        <xsd:restriction base="xsd:string">
                            <xsd:enumeration value="even">
```

```
                    <xsd:annotation>
                              <xsd:documentation>Header will occur on all
even numbered pages.</xsd:documentation>
                    </xsd:annotation>
          </xsd:enumeration>
          <xsd:enumeration value="odd">
                    <xsd:annotation>
                              <xsd:documentation>Header will occur on all odd
numbered pages.</xsd:documentation>
                    </xsd:annotation>
          </xsd:enumeration>
          <xsd:enumeration value="first">
                    <xsd:annotation>
                              <xsd:documentation>Header will occur on the
first page of each section.</xsd:documentation>
                    </xsd:annotation>
          </xsd:enumeration>
     </xsd:restriction>
</xsd:simpleType>
```

Schema definition of Footer element:

```
<xsd:complexType name="ftrElt">
          <xsd:annotation>
                    <xsd:documentation>Footers are areas at in the bottom margin of
each page in the current section.</xsd:documentation>
          </xsd:annotation>
          <xsd:sequence>
                    <xsd:choice maxOccurs="unbounded">
```

```
                <xsd:element ref="aml:annotation" minOccurs="0"
maxOccurs="unbounded"></xsd:element>
                <xsd:element name="cfChunk" type="cfChunkElt"
minOccurs="0" maxOccurs="unbounded">
                    <xsd:annotation>
                        <xsd:documentation>'Context-Free' Chunk
-- allows inline definition of stylesheet, font, and list items</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="p" type="pElt" minOccurs="0"
maxOccurs="unbounded">
                    <xsd:annotation>
                        <xsd:documentation>Paragraph element;
analogous to HTML &lt;p&gt; tag</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
                <xsd:element name="tbl" type="tblElt" minOccurs="0"
maxOccurs="unbounded">
                    <xsd:annotation>
                        <xsd:documentation>Table element;
analogous to HTML &lt;table&gt;</xsd:documentation>
                    </xsd:annotation>
                </xsd:element>
            </xsd:choice>
        </xsd:sequence>
        <xsd:attribute name="type" type="ftrValue" use="required">
            <xsd:annotation>
                <xsd:documentation>Specifies the footer
type.</xsd:documentation>
            </xsd:annotation>
        </xsd:attribute>
</xsd:complexType>
```

FIGURE 5 shows an exemplary flow diagram for representing header and footer information in a ML document, in accordance with aspects of the invention. After start block **510**, the process flows to block **520** where the mini-document information described within a document such as a word-processor document, is determined. The mini-document information used within a document may include many different headers and footers, including those that are not natively supported by later applications parsing the document. Once the mini-document information is determined, processing proceeds to decision block **530**.

At decision block **530**, a determination is made whether a mini-document corresponds to a header or a footer. When the mini-document being examined is a header, processing moves to block **540**. However, if the mini-document is not a header, the mini-document is a footer and processing moves to block **550**. In another embodiment, the mini-document information may describe other structures than headers and footers.

At block **540**, the properties of the header (when the mini-document is a header) are mapped into elements, attributes, and values of the ML file. As an example, the header may include a mini-document that displays the text "confidential" at the top of each page. Three elements that may be used in mapping the properties of a header include the cfChunkElt, the pElt, and tblElt elements (see Table 1). The headers and the properties associated with the headers may change from page to page, section to section, chapter to chapter and the like. There may be more than one mapping, therefore, per document. Once the header properties are mapped, or written to the ML file, processing advances to decision block **560**.

Returning to block **550**, the properties of the footer (when the mini-document is a footer) are mapped into elements, attributes, and values. Exemplary elements used in mapping the properties of a footer include the cfChunkElt, the pElt, and tblElt elements (see Table 1). As previously stated, the footers and the properties associated with the footers may change from page to page, section to section, chapter to chapter and the like. There may be more than one mapping, therefore, per document. After the footer properties are mapped, processing advances to decision block **560**.

At decision block **560**, a determination is made whether all the mini-documents of the document have had their properties mapped to elements, attributes, and values. If not all of the mini-documents have been processed, processing returns to block **530** where the next mini-document is examined to determine whether the mini-document is a header. However, if all the mini-documents have been processed, then the process then moves to block **570**.

At block **570**, the properties of the mini-documents are stored in a ML document that may be read by applications that understand the ML. Once the properties are stored, processing moves to end block **580** and returns to processing other actions.

In another embodiment, the properties of each mini-document are mapped to elements, attributes, and values without a distinction being made between headers and footers.

The above specification, examples and data provide a complete description of the manufacture and use of the composition of the invention. Since many embodiments of the invention can be made without departing from the spirit and scope of the invention, the invention resides in the claims hereinafter appended.